**7. Detailed codes with Relevant comments**

**// code for implementing stack using linked list :**

package stack;

import java.util.Scanner;

public class impstackusingLL {

// Creating node

public class node {

int data;

node next;

node(int data) {

this.data = data;

this.next = null;

}

}

public static int count = 0;

node head = null;

// Function to Push(Add) element in stack

public void push() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter data that you want to push");

int data = sc.nextInt();

node newnode = new node(data);

if (head == null) {

head = newnode;

} else {

newnode.next = head;

head = newnode;

}

count++;

}

// Function to Remove element from stack

public void pop() {

if (head == null) {

System.out.println("Stack is empty");

} else {

System.out.println("Popped element is = " + head.data);

head = head.next;

count--;

}

}

// Function to Read top value in stack without Removing element from stack

public void peek() {

System.out.println("head=" + head.data);

}

// Function to check whether stack is empty of not

public void isEmpty() {

if (head == null) {

System.out.println("stack is empty");

} else {

System.out.println("Stack is not Empty");

}

}

// Function to Count number of elements present in stack

public void size() {

node temp = head;

count = 0;

while (temp != null) {

count++;

temp = temp.next;

}

System.out.println("Count is= " + count);

}

public void display() {

node temp = head;

while (temp != null) {

System.out.print(temp.data + " ");

temp = temp.next;

}

System.out.println(" ");

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

impstackusingLL stk = new impstackusingLL();

int ch;

while (true) {

System.out.println("1.push ");

System.out.println("2.pop ");

System.out.println("3.peek ");

System.out.println("4.isEmpty ");

System.out.println("5.display ")

System.out.println("6.size ");

System.out.println("0.Exit ");

System.out.print("Enter choise = ");

ch = sc.nextInt();

switch (ch) {

case 1:

stk.push();

break;

case 2:

stk.pop();

break;

case 3:

stk.peek();

break;

case 4:

stk.isEmpty();

break;

case 5:

stk.display();

break;

case 6:

stk.size();

break;

case 0:

System.exit(0);

break;

}

}

}

}

**7. Results :->**

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 6

Count is= 1

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 2

Popped element is = 3

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 4

stack is empty

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 0

Process finished with exit code 0

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 5

9 6 3

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 2

Popped element is = 9

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 2

Popped element is = 6

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 3

head=3

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 1

Enter data that you want to push

3

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 1

Enter data that you want to push

6

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 1

Enter data that you want to push

9

**// code for implementing stack using Array:**

package stack;

import java.util.Scanner;

import java.lang.String;

public class stackasADT {

// Intilizing Array of name stack1

int cap;

int[] stack1;

stackasADT(int size) {

this.cap = size;

stack1 = new int[cap];

}

public static int top = -1;

public static int count = 0;

// Function to Push(Add) element in stack

public void push(int data) {

if (top == (cap - 1)) {

System.out.println("Stack is full");

} else {

stack1[++top] = data;

count++;

}

}

// Function to Remove element from stack

public void pop() {

if (top == -1) {

System.out.println("Stack is empty");

} else {

System.out.println("Popped element is = " + stack1[top]);

count--;

top--;

}

}

// Function to Read top value in stack without Removing element from stack

public void peek() {

System.out.println("head=" + stack1[top]);

}

//Function to display elements of stach

public void display() {

for (int i = top; i >= 0; i--) {

System.out.print(stack1[i] + " ");

}

System.out.println(" ");

}

// Function to Count number of elements present in stack

public void size() {

count = 0;

for (int i = top; i >= 0; i--) {

System.out.print(stack1[i] + " ");

count++;

}

System.out.println("Size of stack is = " + count);

}

// Function to check whether stack is empty of not

public void isEmpty() {

if (top == -1) {

System.out.println("Stack is empty");

} else {

System.out.println("Stack is NOT empty");

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of array = ");

int size = sc.nextInt();

stackasADT stk = new stackasADT(size);

int ch;

// For a while loop

while (true) {

//List of choise

System.out.println("1.push ");

System.out.println("2.pop ");

System.out.println("3.peek ");

System.out.println("4.isEmpty ");

System.out.println("5.display ");

System.out.println("6.size ");

System.out.println("0.Exit ");

System.out.print("Enter choise = ");

ch = sc.nextInt();

switch (ch) {

case 1:

System.out.println("Enter data that you want to push");

int data = sc.nextInt();

stk.push(data);

break;

case 2:

stk.pop();

break;

case 3:

stk.peek();

break;

case 4:

stk.isEmpty();

break;

case 5:

stk.display();

break;

case 6:

stk.size();

break;

case 0:

System.exit(0);

break;

}

}

}

}

**7. Results:->**

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 4

Stack is empty

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 6

Size of stack is = 0

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 0

Process finished with exit code 0

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 3

head=14

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 5

14

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 2

Popped element is = 14

Enter the size of array =

5

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 1

Enter data that you want to push

14

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 1

Enter data that you want to push

25

1.push

2.pop

3.peek

4.isEmpty

5.display

6.size

0.Exit

Enter choise = 2

Popped element is = 25

**// code for implementing Queue using Linked list:**

package queue;

import java.util.Scanner;

public class ImpqueusingLL {

class node {

int data;

node next;

node(int data) {

this.data = data;

this.next = null;

}

}

public static int size = 0;

public static node head;

public static node tail;

// Function to add element in queue

public void Enque(int data) {

node newnode = new node(data);

if (head == null && tail == null) {

head = tail = newnode;

} else {

tail.next = newnode;

tail = tail.next;

}

}

// Function to remove element from queue

public void deque() {

if (head == tail) {

head = tail = null;

} else {

head = head.next;

}

}

// Function to check whether queue is empty or not

public void isEmpty() {

if (head == null && tail == null) {

System.out.println("Queue Is Empty");

}

}

// Function to count number of elements in queue

public void size() {

node temp = head;

size = 0;

while (temp != null) {

size++;

temp = temp.next;

}

System.out.println("Size is =" + size);

}

// Function to peek first element of queue

public void peekFront() {

System.out.println("Front element is =" + head.data);

}

// Function to peek last element of queue

public void peekRare() {

System.out.println("Rare element is =" + tail.data);

}

// Main function

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

ImpqueusingLL quell = new ImpqueusingLL();

int ch;

while (true) {

// List of choise

System.out.println("1.Enqueue");

System.out.println("2.Dequeue");

System.out.println("3.Is empty");

System.out.println("4.size");

System.out.println("5.Peek front");

System.out.println("6.Peek rare");

System.out.println("0.Exit");

System.out.print("Enter youre choise = ");

ch = sc.nextInt();

switch (ch) {

case 1:

System.out.print("Enter the data that you want to insert = ");

int data = sc.nextInt();

quell.Enque(data);

break;

case 2:

quell.deque();

break;

case 3:

quell.isEmpty();

break;

case 4:

quell.size();

break;

case 5:

quell.peekFront();

break;

case 6:

quell.peekRare();

break;

case 0:

System.exit(0);

break;

}

}

}

}

**7. Results:->**

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 6

Rare element is =7

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 3

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 0

Process finished with exit code 0

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 2

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 4

Size is =2

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 5

Front element is =56

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 1

Enter the data that you want to insert = 14

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 1

Enter the data that you want to insert = 56

1.Enqueue

2.Dequeue

3.Is empty

4.size

5.Peek front

6.Peek rare

0.Exit

Enter youre choise = 1

Enter the data that you want to insert = 7